RECOMMENDATION SYSTEM FOR FREQUENT SHOPPERS

by

Mert UZGÜL

Engineering Project Report

Yeditepe University

Faculty of Engineering

Department of Computer Engineering

2020

RECOMMENDATION SYSTEM FOR FREQUENT SHOPPERS

APPROVED BY:

Assistant Prof. Dr. Onur Demir (Supervisor)	……………………………….
Prof. Dr. Gürhan Küçük	……………………………….
Assistant Prof. Dr. Funda Yıldırım	……………………………….

DATE OF APPROVAL: /   /2021

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Assistant. Prof. Dr. Onur Demir for his guidance and support throughout my project.

Most importantly, thank you to all of my family for their full support and love. Their presence is very precious to me.

Also I would like to thank my friends Selin Değirmencioğlu and Mert Cenk Timurlenk for their constant source of inspiration and motivation during the project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

UML	Unified Modeling Language
SGD	Stochastic Gradient Descent
GD	Gradient Descent
RMSE	Root mean squared error
MAE	Mean absolute error
P	User-Feature matrix
Q	Product-Feature matrix
R	Original matrix
R^	Predicted matrix
e	Error between predicted and original matrix
b	Bias parameters
L	Latent Features
sgdLearner	Stochastic gradient descent optimization algorithm’s parameter

ABSTRACT

RECOMMENDATION SYSTEM FOR FREQUENT SHOPPERS

Recommendation systems have recently been included in all mobile applications and websites. Since it is a known fact that recommendation systems have a great impact on the user, it has become a part of almost all companies that spend resources on it. This project, together with the matrix factorization methodology, which is a subclass of collaborative filtering technique, will be able to predict the rating users will give to a product that they have not rated before. It aims to optimize the training process with stochastic gradient descent which is an optimization algorithm, and to create a collaborative filtering-based recommendation engine, using methodologies such as matrix factorization, stochastic gradient descent and latent features.

Within the scope of this project, recommendation engine and mobile application will be developed using Python and Flutter programming languages, Firebase database and their libraries. Using open-to-use data sets that published by Amazon between May 1996 and June 2014, the necessary data sets are provided to the recommendation engine for data training.

Recommendation engine is trained with 15000 user ratings. After the training, how accurate recommendations are given to users will be tested and it will be determined how well the system is working. With using root mean squared formula the error rate will be calculated. The margin of error expected to reduce less than three percent to reach goals and successful recommendations. Users can see their recommendations via a mobile application.

ÖZET

SIK ALIŞVERİŞ YAPAN KULLANICILAR İÇİN TAVSİYE SİSTEMİ

Tavsiye sistemleri son zamanlarda bütün mobil uygulamalarda ve web sitelerinde yer edinmiş durumdalar. Bilindiği üzere tavsiye sistemleri kullanıcılar üzerinde oldukça etkili. Şirketler de bu sistemler için oldukça kaynak ayırmaktalar. Bu proje kapsamında, işbirliğine dayalı filtreleme modelinin alt sınıflarından birisi olan matris çarpanlara ayırma metodolojisini kullanarak bir tavsiye sistemi oluşturulması hedef alınmaktadır. Tavsiye sistemi, matris çarpanlara ayırma metodolojisini kullanarak, kullanıcıların daha önce derecelendirmediği bir ürünü, daha önce diğer kullanıcının o ürün için verdiği derecelendirmeleri kullanarak, daha önce derecelendirme yapmamış kullanıcının o ürün için ne tür bir derecelendirme yapacağını tahmin etmeye çalışmaktadır. Bu metodoloji için oldukça sık kullanılan ve bir optimizasyon algoritması olan olasılıksal eğim inişi algoritması ile kullanıcı ön yargıları ve gizli özellikler parametreleri ile birlikte projede tavsiye sisteminin kalbini oluşturmaktadır.

Bu proje kapsamında, tavsiye motoru ve mobil uygulama Python ve Flutter programlama dilleri, Firebase veri tabanı ve onların kütüphanelerinden faydalanarak geliştirilmiştir. Amazon'un Mayıs 1996 – Haziran 2014 tarihleri arasında herkesin kullanımına açık ve gerçek kullanıcılar tarafından derecelendirilen market ürünleri Amazonun sağladığı veri setinin gerekli kısımları kullanılarak tavsiye motorunun eğitimi için kullanılmaktadır.

Öneri motoru, 15000 kullanıcı derecelendirmesi ile eğitilmiştir. Eğitim sonrasında kullanıcılara ne kadar doğru tavsiyelerin verildiği test edilecek ve sistemin ne kadar iyi çalıştığı belirlenecektir. Ortalama karekök hata formülü kullanılarak, hata oranı hesaplanacaktır. Daha başarılı öneriler almak için hata payının yüzde üçten daha az az olması beklenmektedir. Kullanıcılar önerilerini bir mobil uygulama üzerinden görebilirler.

1. INTRODUCTION

Almost everyone knows that recommendation algorithms, which have been hugely popular lately, are in high demand and that companies spend serious resources to develop such systems[3]. In 2006, Netflix organized a $ 1 million contest to improve its own recommendation system in the award-winning competition[11]. Netflix provided a data set containing over 100 million ratings and tested the competitors' work using the root mean square error formula. Contestants who would make the Netflix algorithm provide 10 percent better performance would be awarded. The winner group in this competition used the matrix factorization technique, so that they proved the success of this technique[1]. Thus, matrix factorization methodology has become dominant. Additionally, as a result of this competition, it proved to be better than the nearest neighbor techniques previously used conventionally.

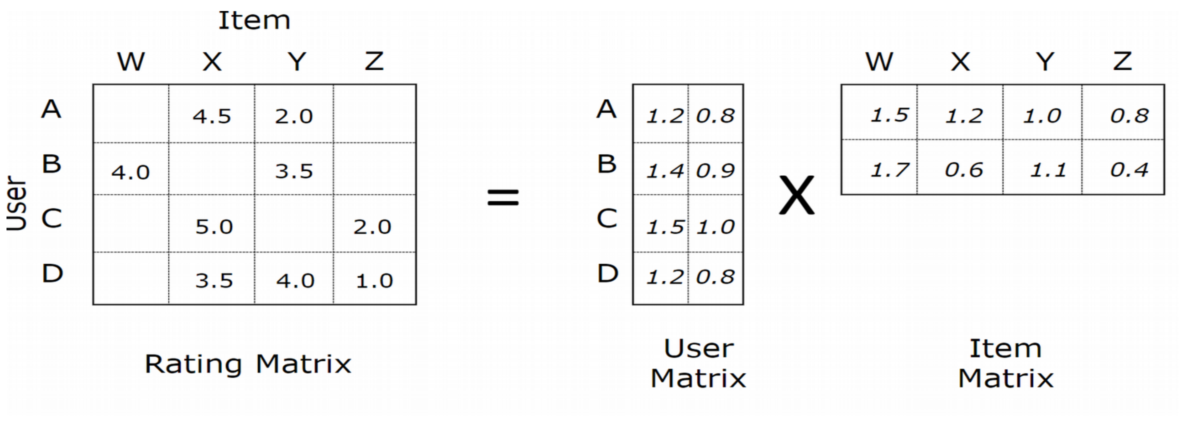


Figure 1.1 Matrix factorization model

As can be seen above, the working method of the matrix factorization method is shown. Starting from this context, started to research how to apply the matrix factorization method in the next objective recommendation engine to my own project.

Recommendation systems are the most popular systems of data science today[3]. In an informal abbreviation for these systems, they are trying to predict what is going to be user's next move. There are programming languages that are using in machine learning algorithms to make recommendation systems.

Lately, Python has become the most popular language in machine learning if we compare with Java, C++. Python is an object oriented, interpretative, unitary and interference high level programming languages[13].

Python libraries have developed and grown so much that they are now preferred by software developers programming in other languages. Extensive resources of libraries on machine learning are being used by data scientists[13]. Python is preferred because it is easy to use, test, and write code. Also with huge communities the language has grew up in a short time therefore, preferred this language to build a recommendation system.

Considering websites like Netflix we see that the moment you watched a movie their recommendation system works immediately. It remembers your choice and provides to user recommended movies.

There are typically 2 different types of recommendation systems in widely use;

- Content based recommendation systems.

-Collaborative Filtering based recommendation systems.

Content based filtering recommendation system basically depends on the similarity between products, user preferences for product features based o their old actions. It has become widely used in many different applications[14].

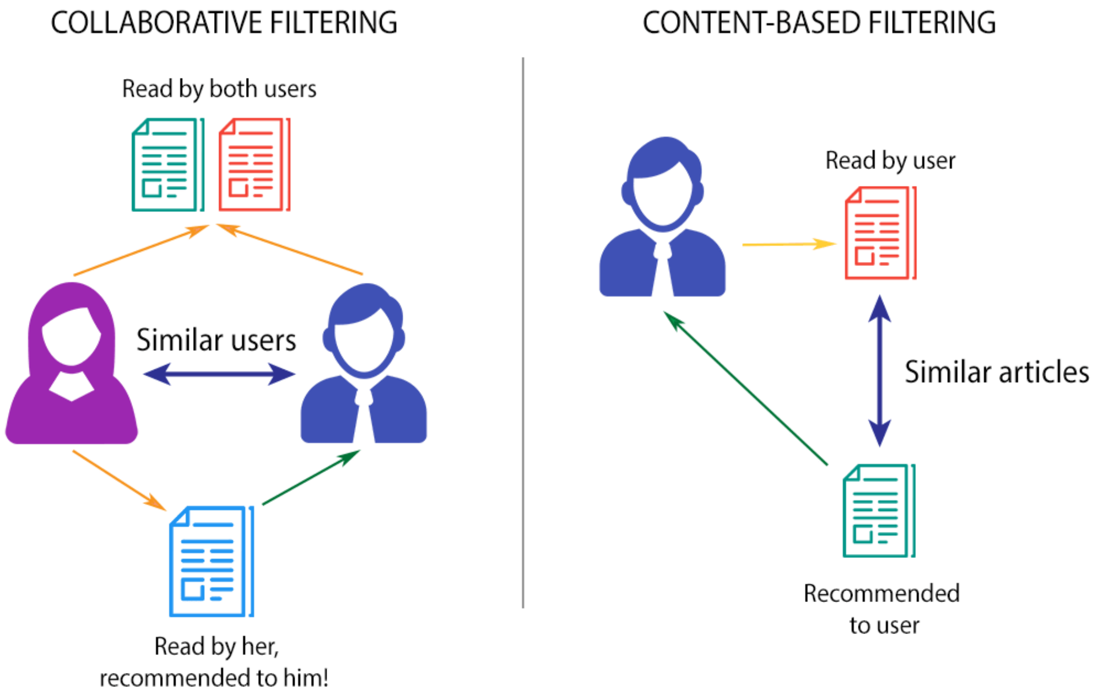


Figure 1.2 Collaborative Filtering and Content based Filtering Model

Collaborative Filtering is a technique in recommendation systems. It is commonly used in well-known companies and applications like Amazon, Netflix, iTunes, IMDB[6]. In collaborative filtering, algorithms are used to make predictions about a user's interests by comparing preferences from several users.

1.1 Problem Definition

Since neural networks, machine learning and deep learning are the most popular and studied topics of recent times[15]. The resources spent on creating recommendation systems showed how important this work was. I had the opportunity to read in some articles that the matrix factorization technique, which is a subject of linear algebra, is used in a way that gives quite successful results while creating the recommendation engine.[1]

Recommendation systems have features that can appeal to people of all ages[3]. For a user to receive advice from a system, it now only takes one click on a site. Most applications and web systems are in a large system that tries to follow and understand everything you do, and instant returns are encountered.

While researching machine learning studies that have been done before, generally systems that give movie recommendations are encountered, started to research on a system that can recommend market products as a system that has been dealt with very little before and decided to develop and train a recommendation engine using real data sets of companies that collects big data such as MovieLens, Amazon and started to work. The most important thing to consider when creating such a system is to be able to give correct recommendations and to have an acceptable margin of error. The work done for this will be explained in detail in the analysis and implementation part of the project.

Netflix's 2006 contest achieved the most successful result with the $1 million award, which was stated to be 10% improvement of the recommendation system[1], matrix factorization method, a linear algebra method used by the winning group, and stochastic gradient descent, an optimization algorithm. Therefore, in the development of the recommendation system within the scope of this project, the matrix factorization technique was used and the error rate was tried to be minimized by stochastic gradient descent.

1.2 Motivation and Aims

Recommendation systems have become an important part of our lives now. In almost every application we use, we come across recommendations on every site we visit. These systems have many features, from companies making more profits to deep learning algorithms that will get to know users better and understand what they like or dislike. These systems are so important now that even $1 million competitions were held on them [9]. In this contest held by Netflix in 2006, the method of matrix factorization, a subclass of the collaborative filtering technique, was the winner of this competition[1]. Based on this context, decided to develop a project on this subject.

Within the scope of this project, decided to create my own recommendation engine using the data Amazon obtained from users between 1996-2014. A recommendation system will be created with the methodologies, techniques and algorithms used in this process. Expecting the estimated values of the products recommended during the project's testing process to have an error rate of less than 3 percent. Hoping to gain much knowledge about recommendation systems and their evaluating metrics and general information about machine learning topic in the end of the project.

2. BACKGROUND

Over the years, the incredible growth of big data created by users on the Internet and the technology age we are in has begun to be fed with recommendation systems to take customer satisfaction to the next level[12]. It has provided an opportunity for companies to analyze and research customer feedback and use them in business organizations. Although such a large amount of data forced the operation of algorithms to improve, it also brought some difficulties. Another issue that needed to be changed and dealt with was the conceptual deviation. Recommendation systems are essentially one of the fastest-changing environments where user interests and related advice witness changes over time for many situational or natural reasons[10]. Products used by users are popular for a certain period of time and then disappear suddenly. Traditional recommendation systems that only take into account the user's past ratings and often disregard differences in the process perform poorly and may give rise to false recommendations. For such reasons, dynamic advice systems began to replace traditional recommendation systems. Dynamic recommendation systems have become a highly researched and studied subject recently. When user information is divided into statically and dynamically, we can see;

Static data are attributes or properties that do not change for the user or can be used to learn the benefit of recommendations that take longer before they change[2]. Examples of this are identity, age, gender and occupation, race can be used as static data to make a personalized recommendation.

Dynamic attributes are the attributes that tend to change as quickly as possible[2]. Examples of these are the characteristics that can change instantly, such as the change in the preference of the users, the change of the environment, the disappearance of the products that have lost their popularity, the change in social relations and seasonal changes. A recommendation system working purely on static data may not produce a recommendation that the user currently needs[10]. Therefore, they create varieties of concept deviations that must be modeled precisely. These will be discussed in the next page in article.

1. -Change in user preferences: A user can change his preferences for products with a no particular reason, so ignoring this when modeling a user's preferences can lead to undesirable recommendations.

2. -Change of product popularity: The perception and popularity of the products by people may decrease after a while so that they can change their preferences. A product can suddenly become popular for external reasons and likewise lose its popularity rapidly.

3. -Main features of the products: Change in item attributes has an equal chance of occurring as user behavior or item popularity changes over time. For example changing a computer's content, technology and performance can increase interest in that product when compared with old one.

4. -Dynamic Interest in the Community: Users who like the same products at the same time may tend to like other products after a while. Likewise, the interest of communities may change. Therefore, it would be useful to handle such issues when setting up recommendations.

5. -Changes in season: Seasonal changes cause the users to change their choice of products. It should be avoided to recommend a seabed to a user in winter and a coat in summer if there is any special offer. The aim here is to emphasize the importance of giving advice to users according to the seasonal features.

6. -Changes in user bias: It is an important point to focus on the change in user bias. The reason for bias arises when some users rank higher than others who tend to rate lower, when other users rank some items higher than others. For example, a user who scores 4 points for an average-tasting food order may tend to give 3 points the next time the same food order.

7. -Temporary and short term user changes: A user may give a very low rating to a product although he previously rated very well to a product according to the current mood on some days. Such effects generally do not last more than a few days.

8. -Varying social relationships: Changes in people's social environment can cause big changes in their product preferences.

2.1. Relational work

Recommendation systems are defined as a decision-making mechanism provided to users in mixed information environments [2]. At the same time, these systems try to give recommendations based on the choices of other users in order to make it easier for the users to make a choice [3]. Many methods have been developed to create such recommendation systems [4]. Collaborative filtering technique is one of the most widely used. This technique identifies users with similar tastes and recommends products to them. Companies such as Amazon, Netflix, Spotify, GroupLens take advantage of these techniques [1]. There are articles and researches about Amazon. Amazon is using topic diversification algorithms to provide better recommendations for customers. For example, this system uses a product x product matrix to create a table of similar products and use a collaborative filtering technique to overcome the scalability problem. As a different example, Netflix's personalized recommendation algorithms generate $1 billion a year from customer retention [9]. The recommendation systems in Netflix cover various algorithmic approaches such as deep learning, neural networks, causal modeling, probabilistic graph models, matrix factorization[9]. Along with these algorithmic approaches, Netflix's recommendation systems calculate the rate of users watching a certain content by analyzing millions of users based on multiple factors[11].

Several drawbacks have been established, despite the success of these two filtering techniques. Restricted content analysis, overspecialization and sparsity of data are some of the issues associated with content-based filtering techniques. Collaborative methods also show issues with cold-start, sparsity and scalability. These problems usually reduce the quality of recommendations. In order to mitigate some of the problems identified, Hybrid filtering, which combines two or more filtering techniques in different ways in order to increase the accuracy and performance of recommender systems has been proposed.

3. METHODOLOGY

In this section, information will be given about the methodologies underlying the project. Methodologies are matrix factorization, stochastic gradient descent, latent features, and bias. Each of them will be explained in the below.

3.1 Matrix Factorization

Some of the most successful realizations of latent factor models are based on matrix factorization[1]. Matrix factorization characterizes both items and users as follows: Vectors of factors extracted from item rating models. Similar correspondence between item and user factors leads to build a recommendation. These methods have become very popular by combining high scalability with predictive accuracy in recent years[10]. Additionally, they offer a lot of flexibility to model a variety of real life situations. Strength of the matrix factorization is not needed to know too much information about the user[1].

Mathematics behind the matrix factorization is quite understandable[1]. Suppose we have item i associated with a vector $q_i \in R^f$ and the user u associated with a vector $p_u \in R^f$, resulting dot product, $q_i^T p_u$, shows the interaction between user and item, the user's total interest in the item's characteristics. As a result of these estimates can be represented with r_{ui};

Equation 3.1 $$\hat{r}_{ui} \sim q_i^T p_u$$

The biggest challenge here is computing the mapping of each item and user to factor vectors[1]. After computing, we become able to make the recommendation system according to the 1st formula, and we can estimate the user's ratings.

3.2 Stochastic Gradient Descent

It may be necessary to know what the gradient descent is before beginning the stochastic gradient descent. Gradient descent is a commonly used optimization technique, such as machine learning, and gradient can, in fact, be declared as a function slope. In conjunction with the shift of variables, it calculates the degree of change of other variables.

The stochastic gradient descent algorithm is used for optimization[1]. The algorithm randomly picks samples in every iteration from data set and after training it starts to estimate the ratings given by users and gives the margin of error.

Equation 3.2 $$e_{ui} = r_{ui} - q_i^T p_u \text{ [2]}$$

One of the main reasons the stochastic gradient descent algorithm is used instead of gradient descent is the need to train an entire data set at once in gradient descent[16]. For this reason, problems arise when the data sets are enlarged, for example the computational cost becomes very high. The reason for using stochastic gradient descent is preferred because it allows you to make each iteration piece by piece instead of doing it all at once. In stochastic gradient descent technique, only one sample is chosen from the data set for the iterations, the way it took by the algorithm to reach the minimum is noisier than gradient descent algorithm. This is not so crucial because the way it took by the algorithm is do not considered at all because there is way, with the minimum and the remarkably shorter training time.

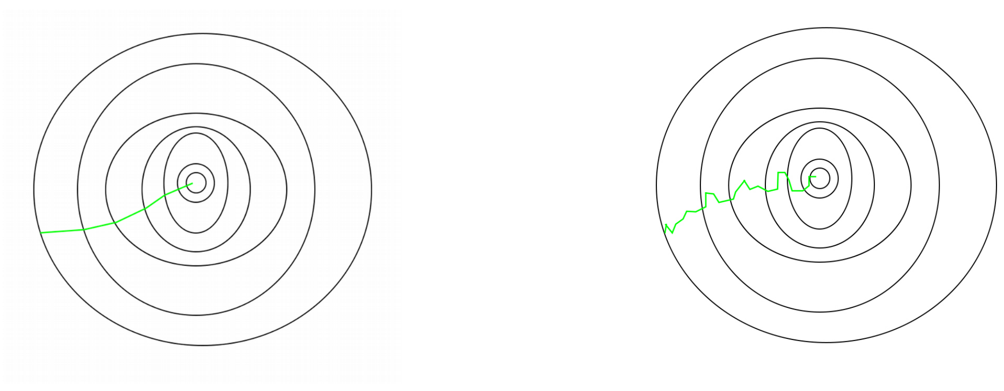


Figure 3.1 Gradient descent versus Stochastic gradient descent

3.3 Bias Parameter

It is shown that bias is typically introduced as a parameter in the matrix factorization model. The user's bias indicates that the user chooses to offer higher or lower ratings than normal, to illustrate why bias is used[1]. For instance, the bias of a film will explain how well this film is scored relative to the average. This depends only on the video, and the connection between a consumer and the film is not taken into account. b_{ui} and accounts for the consumer and item effects denote the bias involved in ranking r_{ui}. The ranking is denoted by μ; the b_u and b_i parameters indicate user u and item i respectively.

Equation 3.3 $$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

3.4 Latent Features

The task of estimating missing ratings can be thought of as filling in the gaps so that the values are consistent with existing evaluations in the matrix[1]. There's a hunch behind matrix factorization to rate unrated products. There should be some latent features that determine how the user will rate a product. If these latent features are discovered, we can estimate the rating a user gave to a product. This is because the user-related features match the product-related features and a degree of recommendation emerges.

Equation 3.4 $$\hat{r}_{ui} = q_i^T p_u = \sum_{L=1}^{L} p_{uL} q_{Li}$$ [1]

4. ANALYSIS AND DESIGN

In the analysis section functional and non-functional parts will be discussed and listed. On the other hand in the design section UML diagrams and figures will be shown.

4.1 Functional Requirements

Functional requirements gives details about the system in other words functional requirements describes what system does.

As explained topics above, in this graduation project, with using matrix factorization method i am trying to build a recommendation system that predicts what kind of evaluation users could make about the products that they did not evaluate yet. Recommendation system is integrated with a mobile application for usability purposes in other words application has been developed where users can easily see the recommendations given to them. While planning the project idea in the first place, There was an idea to design an application where they can visually see the recommended products on the mobile application, but since the product names are not shared in the data set provided by Amazon, the products recommended to the user will be displayed using the product ID. This situation may decrease user satisfaction.

When a user starts the application s/he will meet with login screen. User must enter his user id. After s/he enters the user id they can click to the login button to reach next step. Firebase database management system is stores the user id’s, product id’s and the ratings. All these data can be seen by the admin in the database. After user clicks to login button with his user id, recommended items list will be shown to user. In this part user can see the products that are recommended to him. If there is a new user or item trained matrix will be updated.

4.2 Non-Functional Requirements

Non functional requirements describes the quality features of the software. Each feature in this project will be given below.

- Error rate

The error of the process is called the inaccuracy of expected output values[10]. The error is expressed as an error rate when goal values are categorical. The error rate in a recommendation system should be within acceptable levels. Using stochastic gradient descent algorithm error rate will was reduced to very low levels. Within the scope of this project, it is possible to see a margin of error approaching to zero percent when small data sets are used, and when large data sets are used, it has been seen to drop below three percent depending on the number of iterations.

- Usability

Features such as consistency and aesthetics in the user interface are assessed by usability. Within the scope of this project trained recommendations are displayed in a user-interface supported mobile application for easy access and satisfaction of users.

-Scalability

In response to changes in application and system processing requirements, scalability is the measure of the ability of a system to increase or decrease performance and cost. In real world some data sets are up to 5 TB and their training will be quite difficult. It will not be possible to train the extremely huge data sets in an individual computer it needs much more powerful systems to train them[10]. Within the scope of the project Amazon’s grocery product data set which is up to 1 million user, is decreased to fifteen thousand train in an individual computer.

-Reliability

Reliability is an evaluation criteria metric that is also used in a machine learning topics. Like all software applications, the reliability of machine learning systems is frequently linked to the system's error tolerance and recoverability in producing. In fact, the reliability of ML systems is linked to how reliable the training process of ML models is. Further, works are considered in individual machine learning about reliability in[5]. The reliability of a machine learning model can be evaluated, training and testing with their error rates and data set quality and evaluating models. Within the scope of this project after researching the details and learning techniques that are used frequently and successful decided to work with Amazon’s data set and matrix factorization technique with stochastic gradient descent algorithm. It is crucial to work with real data sets for the value of work.

-Fairness

In machine learning, an algorithm used should be fair. It is the separation of variables, especially the characteristics of individuals who are thought not to be related to the outcome, from those that are considered sensitive in this regard. For example, the fact that the user is independent of gender, disability, ethnic origin and sexual orientation in the system indicates that a recommendation system is fair. This is one of the important reasons why using matrix factorization because the matrix factorization technique inherently does not require such user features when used in the recommendation system.

4.3 Design

4.3.1 Requirement Specifications of the System

Use case diagrams are the diagrams that represent the user's interaction with the system[17]. Using these diagrams, what the users can accomplish in the system is shown. In this way, we get the opportunity to examine the situation of the user and administrator. In the first diagram, the user-application relationship will be shown, and in the second diagram, we will examine the relations of the administrator in the application, database and data training process.

In the first diagram the only actor is the user. As soon as the user enters the application, they will see a login screen. There are two different options the user can make here. If the user already has an id, they can enter their id to move on to the next step of the application. If the user does not have an id, he or she needs to register to the system and for this he will need to click a register button and create a user id.

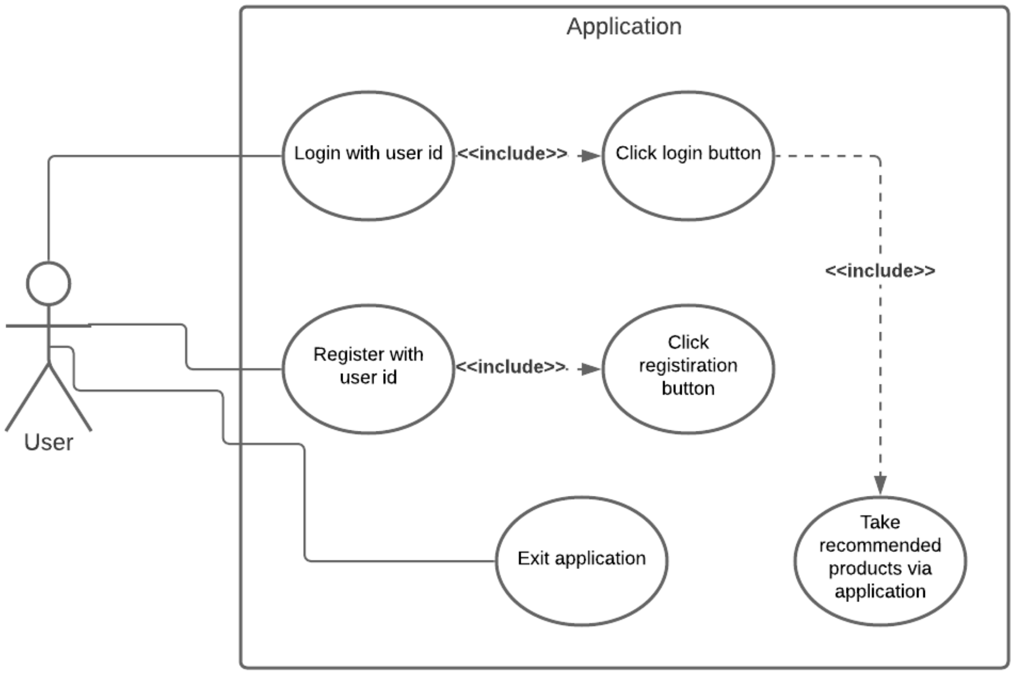


Figure 4.1 Analysis of the user's scenario

Second diagram will be about the system administrator. The role of the manager in the system is to follow the events and fulfill the requirements. Since all user data is kept in the Firebase database management system, when a new user registers with the recommendation system, it will be automatically registered in the database. Administrator does not need to do registrations manually. After the user registers to the system, the recommendation algorithm will be trained and the new user will be able to use the recommendation system.

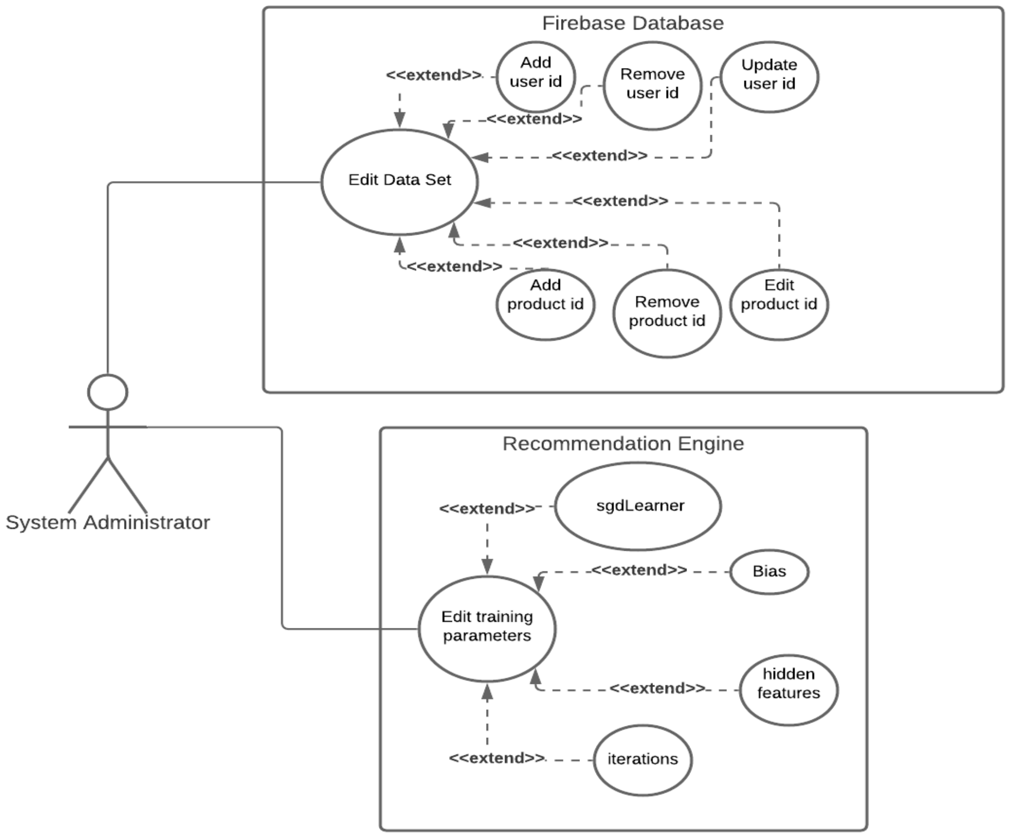


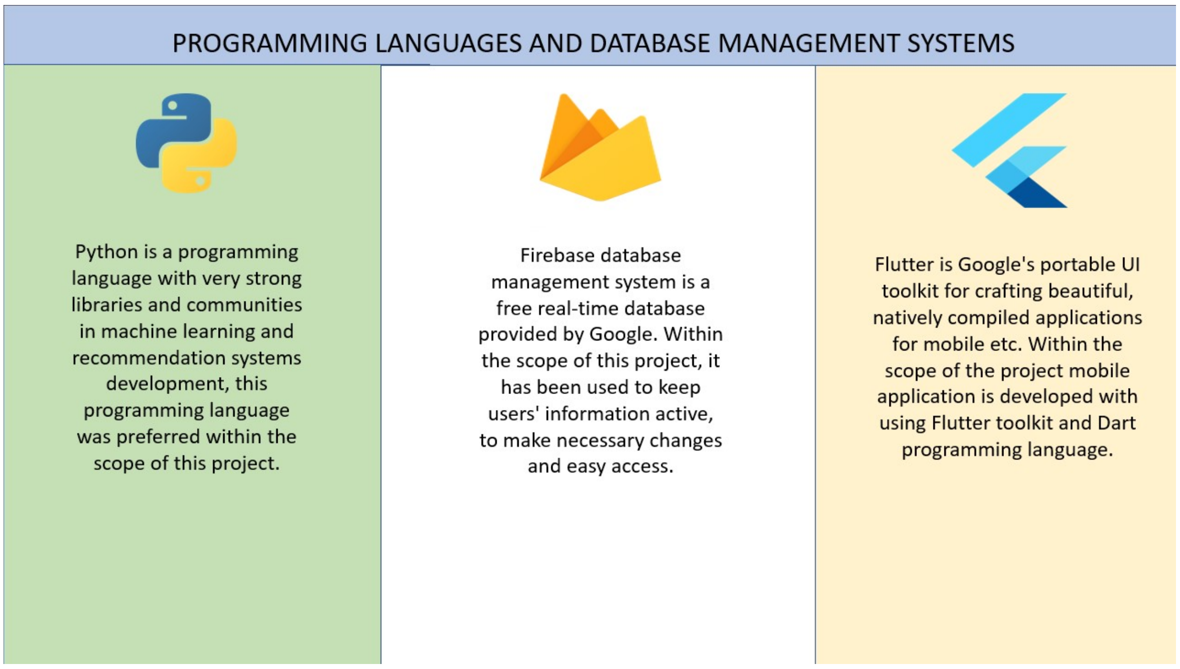
Figure 4.2 Analysis of the administrator’s scenario

5. Implementation

In this topic of the project implementation details, overall architecture and application will be shown with snapshots obtained from recommendation engine, application and Firebase database management system.

5.1 Overall architecture

The project consists of 3 components that are required by its nature. These are respectively mobile application, recommendation engine and database management system. The implementation steps will be explained sequentially and visualized in detail. There is a figure below that represents tools, programming languages and database system visually.



PROGRAMMING LANGUAGES AND DATABASE MANAGEMENT SYSTEMS

Python is a programming language with very strong libraries and communities in machine learning and recommendation systems development, this programming language was preferred within the scope of this project.

Firebase database management system is a free real-time database provided by Google. Within the scope of this project, it has been used to keep users' information active, to make necessary changes and easy access.

Flutter is Google's portable UI toolkit for crafting beautiful, natively compiled applications for mobile etc. Within the scope of the project mobile application is developed with using Flutter toolkit and Dart programming language.

Figure 5.1 Programming languages and Database management systems

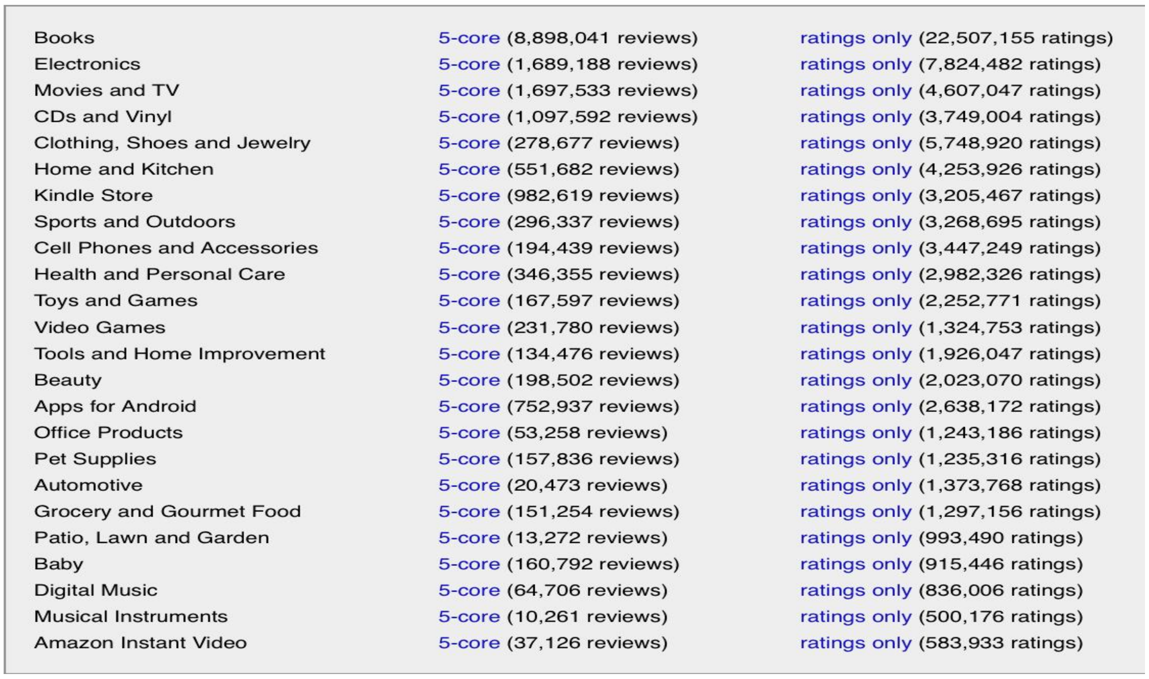
5.2 Phases of implementation

First of all, in order to proceed to the implementation phase of the project, it is necessary to start from long-term research and article reviews, how and according to what the recommendation system will be created, which libraries will be used, what parameters will be included in the system and the correct data set. In the next topic data set collection will be discussed.

5.2.1 Collection of Data

When creating a recommendation system, it is necessary to collect data as a starting point. Data can be gathered in two different forms, explicitly or implicitly. The explicit data consists of users' comments about the products and their ratings. On the other hand, implicit data consists of user history, search history and pages viewed, and so on, unlike explicit data. Users tend to have different tastes about products so the data set reaches a distinct structure. Feeding recommendation systems with more data throughout the process leads to smarter recommendations and mediates users to buy those products.

Within the scope of this project data is gathered from Amazon’s open-to-use data set collection spanning May 1996-July 2014[12]. The main reason for using Amazon's data set was that it consisted of over 142 million comments and metadata, all of which were data from real users. Due to the working principle of the matrix factorization technique, a rating-based data set was needed and I reached it using Amazon's data sets. Thus, one of the first and most important steps in the implementation of the project was completed. On the page below there is a figure showing these data sets.



Books	5-core (8,898,041 reviews)	ratings only (22,507,155 ratings)
Electronics	5-core (1,689,188 reviews)	ratings only (7,824,482 ratings)
Movies and TV	5-core (1,697,533 reviews)	ratings only (4,607,047 ratings)
CDs and Vinyl	5-core (1,097,592 reviews)	ratings only (3,749,004 ratings)
Clothing, Shoes and Jewelry	5-core (278,677 reviews)	ratings only (5,748,920 ratings)
Home and Kitchen	5-core (551,682 reviews)	ratings only (4,253,926 ratings)
Kindle Store	5-core (982,619 reviews)	ratings only (3,205,467 ratings)
Sports and Outdoors	5-core (296,337 reviews)	ratings only (3,268,695 ratings)
Cell Phones and Accessories	5-core (194,439 reviews)	ratings only (3,447,249 ratings)
Health and Personal Care	5-core (346,355 reviews)	ratings only (2,982,326 ratings)
Toys and Games	5-core (167,597 reviews)	ratings only (2,252,771 ratings)
Video Games	5-core (231,780 reviews)	ratings only (1,324,753 ratings)
Tools and Home Improvement	5-core (134,476 reviews)	ratings only (1,926,047 ratings)
Beauty	5-core (198,502 reviews)	ratings only (2,023,070 ratings)
Apps for Android	5-core (752,937 reviews)	ratings only (2,638,172 ratings)
Office Products	5-core (53,258 reviews)	ratings only (1,243,186 ratings)
Pet Supplies	5-core (157,836 reviews)	ratings only (1,235,316 ratings)
Automotive	5-core (20,473 reviews)	ratings only (1,373,768 ratings)
Grocery and Gourmet Food	5-core (151,254 reviews)	ratings only (1,297,156 ratings)
Patio, Lawn and Garden	5-core (13,272 reviews)	ratings only (993,490 ratings)
Baby	5-core (160,792 reviews)	ratings only (915,446 ratings)
Digital Music	5-core (64,706 reviews)	ratings only (836,006 ratings)
Musical Instruments	5-core (10,261 reviews)	ratings only (500,176 ratings)
Amazon Instant Video	5-core (37,126 reviews)	ratings only (583,933 ratings)

Figure 5.2 Amazon's open-to-use data sets

The data set used within the scope of the project is grocery and gourmet foods. Amazon provides close to 1.3 million data available for use in this data set. Although Amazon provides this large data set, it is not possible for individual computers to train such a large amount of data, so about fifteen thousand of this data set has been used. The next issue after data collection will be how these data will be stored.

5.2.2 Storing the data

Another issue that should be taken into consideration with the recommendation systems that provide more successful results with the growth of the data set is how to store these data. One of the first ideas that comes to mind is to use a database system to store data according to the type of data set. For this reason, the Firebase database management system that offered free of charge by Google, began to be used to store data. In the screenshot below, a figure showing the data sets stored in the database is shared.

5.2.3 Analyzing the data & Data cleaning

Before starting the training of a data set, it may be necessary to organize some data in the data set content on a project basis. There are four different primary keys in the ratings-based data set provided by Amazon. These were the user name, product name, rating and time stamp, respectively. Since the matrix factorization technique used within the scope of this project does not require a time stamp, it had to be cleaned from the data set. After cleaning it, using a matrix factorization technique, we have the three primary keys needed to create it. Secondly, the data set had to be minimized in order to be extremely large and to be able to train, to enable the use of Python libraries and to allow individual computers to perform the data training process in an acceptable time frame. For this reason, it was decided to use about fifteen thousand of the data set, which is approximately 1.3 million originally. After the data sets were edited and made sure that they were available for training of the recommendation system, they could be passed to the data analysis part.

Before filtering the data set that will be explained in the next section, it may be necessary to examine the ready data set we have. There are different techniques used in such examinations. Below are some graphic analysis and figures such as heat maps. These figures are provided by Jupyter Notebook.

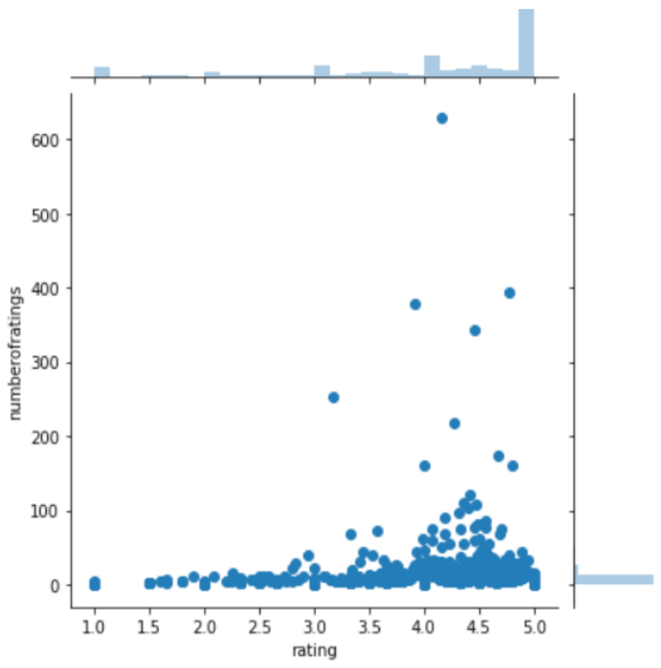
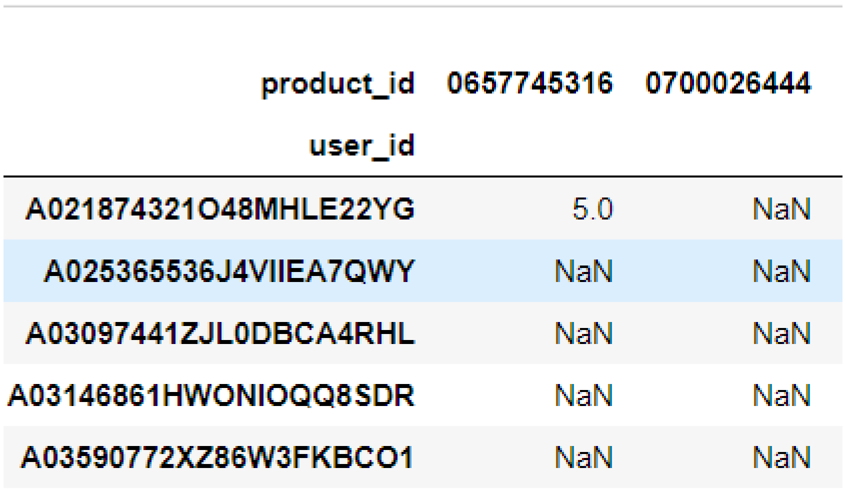


Figure 5.3 Graph of average ratings and number of ratings for products

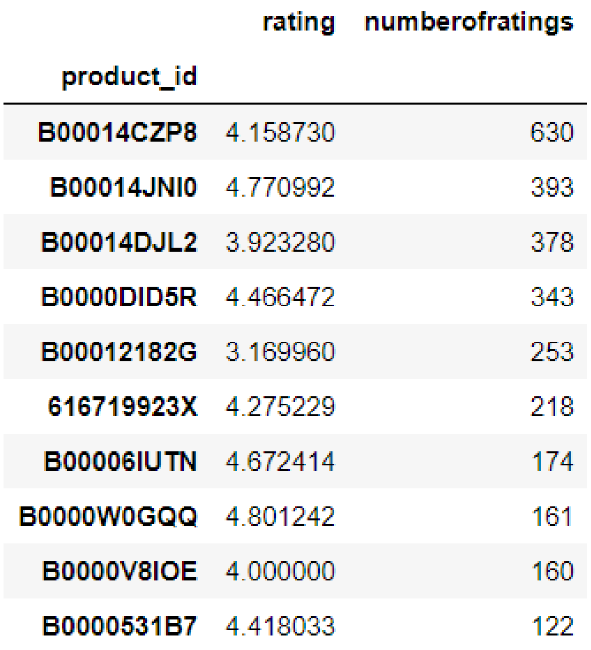
In the graph 5.3, we can see how much rating users gave to which products. Almost some products have only rated once. On some products, it is possible to see more than 600 ratings. Generally users tend to give products 4 or higher ratings up to 5. It is quite possible to understand this with the clustering on the chart.



product_id	0657745316	0700026444
user_id		
A021874321O48MHLE22YG	5.0	NaN
A025365536J4VIIEA7QWY	NaN	NaN
A03097441ZJL0DBCA4RHL	NaN	NaN
A03146861HWONIOQQ8SDR	NaN	NaN
A03590772XZ86W3FKBCO1	NaN	NaN

Table 5.1 Status of the data set before the training process

The chart shared in previous page shows the part of the ratings given by users to certain products before training. Since most users have not given any ratings for most products before, it is quite possible to encounter the statement that this is not a number. This is quite normal since the nature of a rating-based recommendation system is expected. The statement is not a number assumes that the user has not made any ratings, that is, he has not used that product before. When the data set training process starts, it will be replaced by the zero number since the algorithm cannot understand the expression of not a number. In other words, zero means that the user has not made any rating about this product.



	rating	numberofratings
product_id		
B00014CZP8	4.158730	630
B00014JNI0	4.770992	393
B00014DJL2	3.923280	378
B0000DID5R	4.466472	343
B00012182G	3.169960	253
616719923X	4.275229	218
B00006IUTN	4.672414	174
B0000W0GQQ	4.801242	161
B0000V8IOE	4.000000	160
B0000531B7	4.418033	122

Table 5.2 Product-Rating table

The chart above shows the number of times a product was rated by the users before the data set training and the average of the ratings given. By examining charts like these and others, it is possible to see what type of ratings users have on average for a product. Filtering process will be covered in the next topic.

5.2.4 Filtering & Training Data

The data filtering process became ready to start after the data set passed up to this step. In this part of the project, as mentioned in the previous sections of the report, collaborative filtering model and matrix factorization technique will be used to build recommendation engine. We will try to understand which users have similar tastes and recommend products that they will like each other in the future.

First of all, if we need to give some example of matrix factorization, there are 3 different primary keys we will use, which are user, product, and rating. We have a data set that users evaluate in the range of 1-5. A value of 0 means that it has not been evaluated before. We will try to guess what the 0 ratings here might actually be. Using the matrix factorization technique, we will try to find latent features about how users can rate products. First of all, we break the matrix into constituent parts so that the product of these parts forms the original matrix.

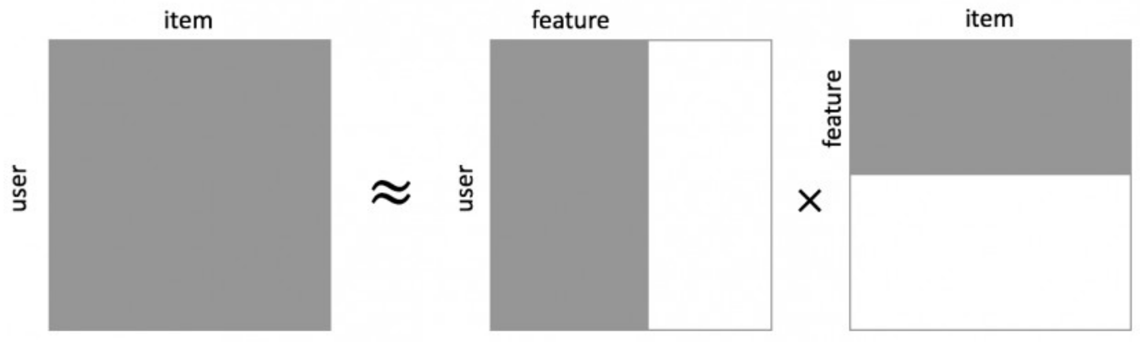


Figure 5.4 Fragmented form of original matrix

An intensive mathematical process begins from this point on. The sections mentioned in the methodology section will be the most important parts of the formation process of the recommendation system. In this section, each algorithm and techniques used step by step until the end of the section will be examined in detail during the training process.

First, let's assume that there are U users and I items (products). Suppose the size of our matrix is R. We will try to find features which are latent, L (latent features). For this we can obtain a new approximate matrix R with (| U | x L) = P and (| I | x L) = Q. It was mentioned in the methodology section that selecting the latent features eliminates the noise in the data. This was to remove features that did not determine how a user rated a product. It was to calculate the dot product of these vectors and obtain ratings based on hidden features.

The equation will be;

Equation 5.1 => $$\mathbf{R \sim P \ x \ Q^T = R^\wedge} \quad [1]$$

(| U | x L) = P shows the relationship between a user and features. Likewise, (| I | x L) = Q shows the relationship between an item and its features. We encounter the step of calculating the dot product of two vectors which are q_i and p_u to estimate the rating about the product.

Equation 5.2 => $$\mathbf{r_{ui} = p_u{}^T q_i = \sum^{L}{}_{L=1} p_{uL} q_{Li}} \quad [2]$$

At the present stage, the process of finding a way to find P and Q begins. To approach this problem, it is necessary to internalize the two matrices with some values and try to understand how different the products are, and try to minimize this iteratively. In fact, this method is the stochastic gradient descent that we talked about in the methodology section. This method tries to find the minimum of the differences. The difference here will give us the error between the actual rating and the estimated rating and calculated for each user-product by the following equation;

Equation 5.3 => $$e^2_{ui} = (r_{ui} - \hat{r}_{ui})^2 = (r_{ui} - \sum^{L}_{L=1} p_{uL} q_{Li})^2$$

After obtaining this equation, it will be necessary to explain what some values are.

- e^2_{ui} = The letter e indicates the error in the equation. The reason for squaring the error is that the predicted ratings may be higher or lower than the actual ratings.

- r_{ui} and $\hat{r}_{ui}$ = The actual rating given to the product by users and the rating that will be estimated by the recommendation system for the product, respectively.

The main goal at this point is to minimize the errors in the P and Q matrix. In order to get the least error, P and Q matrices need to be updated to get optimized versions. This is where stochastic gradient descent comes in. The update below defines it as the gradient of the error to be minimized.

Before proceeding with the equation, it may be necessary to explain what the sgdLearner parameter is and why it is used. The sgdLearner parameter is the learning rate that decides the size of each update. These updates are iterated until the error rate is minimized to acceptable levels. Within the scope of this project, it was repeated 200 times in order to reduce the error rate below 3%. The name of the sgdLearner parameter will replace the letter “a” in the equation.

The equations (both user and product) will be;

Equation 5.4 => $\mathbf{a / a_{p_{uL}} * (e_{ui}^2) = -2(r_{ui} - r^{\wedge}_{ui})q_{Li} = -2e_{ui}q_{Li}}$

Equation 5.5 => $\mathbf{a / a_{q_{Li}} * (e_{ui}^2) = -2(r_{ui} - r^{\wedge}_{ui})p_{uL} = -2e_{ui}p_{uL}}$

Updating can be done for p_{uL} and q_{Li} as they now have gradients. So the following equations will show the updated version.

Equation 5.6 => $\mathbf{p'_{uL} = p_{uL} - sgdLearner * a / a_{p_{uL}} * (e_{ui}^2) = p_{uL} + 2e_{ui}q_{Li}}$

Equation 5.7 => $\mathbf{q'_{Li} = q_{Li} - sgdLearner * a / a_{q_{Li}} * (e_{ui}^2) = q_{Li} + 2e_{ui}\,p_{uL}}$

After these equations were completed, we optimized the P and Q matrices to predict the ratings. The functioning of the algorithm and how matrix factorization works are briefly summarized in below the page and on the following page.

```
# for i in range = 1,2,...., endOfLatentFeatures
    # for each $r_{ui}$ ratings ε R (product-rating matrix) :
        # predict ratings
        # update P and Q matrices
```

Figure 5.5 Pseudo code of matrix factorization prediction

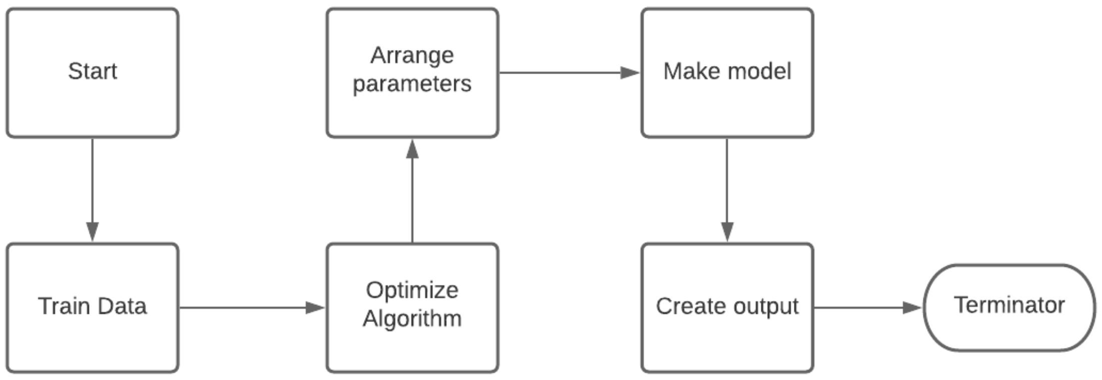


Figure 5.6 Optimization of missing ratings by stochastic gradient descent algorithm

5.2.5 Testing trained data

The training process of the data was examined in detail in previous pages. If we come to how this process is evaluated, Many metrics can be used to evaluate a recommendation system[10]. These evaluation metrics used in recommendation systems are quite significant. Before entering the evaluation process of this project, it will be useful to discuss roughly about the evaluation metrics used in recommendation systems. Below are a few of the most commonly used evaluation metrics.

-Root mean squared error & Mean absolute error(RMSE&MAE):

Root mean squared error is one of the most popular and common used evaluation metric that widely used in recommendation systems[1]. The mean absolute error is also a metric that is widely used too. There are serious studies and articles on these two evaluation metrics on which one is better or not[2].

Although it is used frequently in both metrics, i preferred to use the root mean squared error metric within the scope of this project. Root mean square error measures error in predicted ratings. To briefly explain the metric;

$$RMSE=\sqrt{\frac{\sum_{i=1}^{N}(Predicted_i-Actual_i)^2}{N}}$$

Equation 5.8 Formula of RMSE

After examining the RMSE formula, a question may come to mind. This question may be that a product that the user has not rated before has no actual value and what will be the actual value should be in the formula. The answer to this question is actually hidden in the technique of matrix factorization[8]. Users can make 1 to 5 evaluations about a product. The real value of a product that a user has not evaluated before is initially accepted as 0 also in the Python code missing values is filled with “fillna” method, and 0 is written instead of the actual value in the formula. The actual value of previously unrated products is considered 0 to provide multiplication and the training process begins.

The rating estimated for this formula is obtained by subtracting from the original ratings from the rating estimated by the recommendation system and squaring it, iterating it by the number of ratings up to N, dividing it by the total number of original ratings and taking the square root of all. To give an example of this, let's assume that a user gives a 5 rating to a product, and if the recommendation system predicts the user rating as 4, the root mean square error is 1. The lower the root mean square error, the better the predictive success of the recommendation system.

Within the scope of this project, the root mean square error will be found in every 20 iterations during the data set training and this iteration will be repeated 200 times. With the stochastic gradient descent, it will be optimized at every iteration. In the process, it can be seen in the graph below that the root mean square error has parabolically decreased and gave really satisfactory results[10].

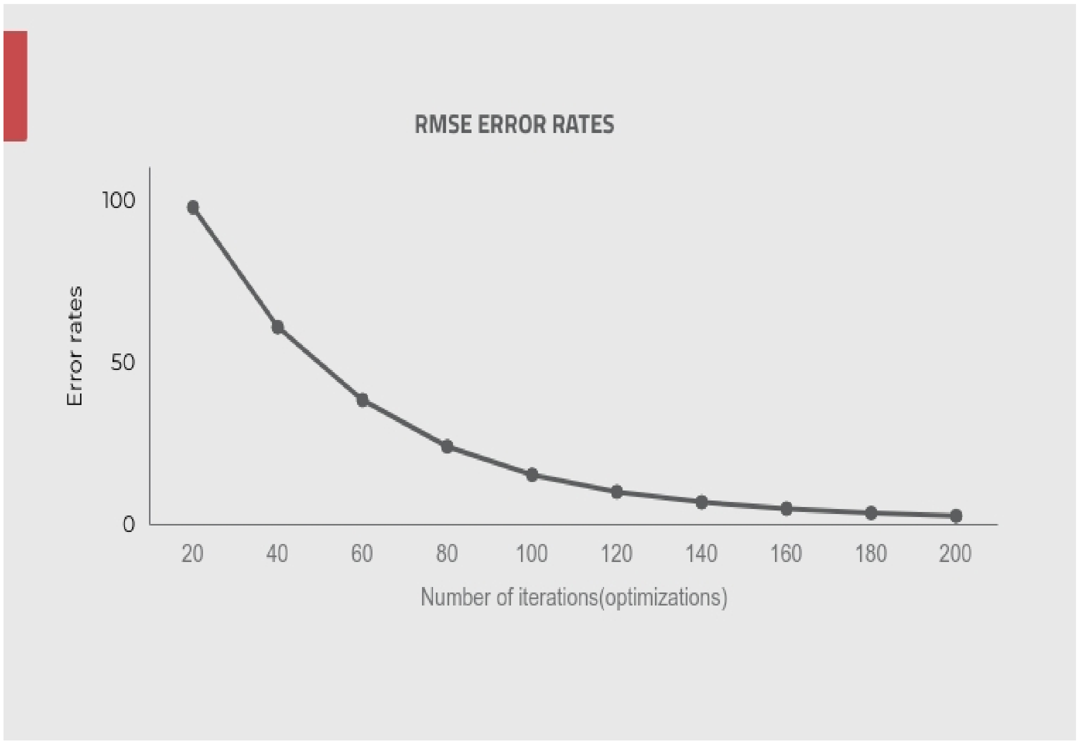


Figure 5.7 RMSE error rates for every 20th iteration in SGD

As can be seen from this graph, it is possible to observe that the error rate decreases parabolically as the number of iterations increases. The reason for this reduction is made by the stochastic gradient descent algorithm as mentioned earlier. Stochastic gradient descent tries to minimize error by making continuous iterations[16]. More iterations means lower error rate, but as expected, more iterations means more costs. The missing values in the rating matrix, that is, the values that have not been graded before, are filled with the r_{ui} values we estimate and then completed by iteratively using the stochastic gradient descent for optimization. The graph in previous page consists of 14008 unique users(some users rates more than one time) giving 15000 unique ratings to 2614 unique products. If we create a matrix of 5 different users and the ratings given by these users to 5 different products to show how well the recommendation system works in another way, we do a test to see if the users who have not rated a product before are actually given the right recommendations.

user_id	product_id	rating
1	1	5
1	2	1
1	3	3
1	4	5
2	1	5
2	2	2
2	3	3
2	5	5
3	1	1
3	2	5
3	5	1
4	5	5
5	2	4
5	3	5

Table 5.3 Created user ratings for test

Due to the structure of collaborative filtering[8], it will provide recommendations using users who have similar ratings. If we look at the chart above, it is possible to visually understand that 1st and the 2nd user have similar tastes. Likewise, the 5 rating given by the 4th user only to the 5th product may indicate that he is a user similar to the 2nd user. In a different way, the 3rd user rated 5 the 2nd product. Also 3rd user rated 1st and 5th product with rating 1. In the end of the training we expect that the recommendation system gives us similar results as we discussed.

	Product 1	Product 2	Product 3	Product 4	Product 5
User 1	4.99	1.02	2.99	4.99	5.10
User 2	4.98	1.99	3.00	5.00	4.98
User 3	1.01	4.98	3.77	2.25	1.01
User 4	4.77	2.94	4.14	5.07	4.99
User 5	4.04	4.00	4.98	4.62	4.38

Table 5.4 Prediction of missing ratings

In the chart above, the ratings given by the recommendation system to products that have not been graded before are indicated in yellow. After examining the chart, another question may come to mind. We said that the ratings would be in the range of 1 and 5, but for the 1st user above, the rating we predicted for the 5th product is greater than 5, how is this possible? Actually, the reason for this was explained in the mathematical process. We're handling the squared error because the estimated rating may be higher or lower than the actual rating.

Another question might be how the latent features parameter works. The effect of latent features on the estimation process can be explained as follows. There should be some hidden attributes that determine how a user rates an item. For example, when the number of features is 2, the algorithm can associate users and items with two different traits, and the predictions follow those relationships. Another point to note is that the total latent features should not be more than the original features and should not be 0. For example, we assume that there are no more than 5 latent values in a 5 x 5 matrix. Because it doesn't make sense to assume that each user is associated with a nonexistent feature. However, it is possible to make such an assumption, but the recommendations given in this case will not make sense.

In the previous chart, before this process, we explained what kind of ratings the recommendation system can make, more precisely what kind of result we expect. Since the results came out as expected, and while doing this training, the root mean square error, along with the stochastic gradient descent algorithm, dropped from 5% to 0.4%. According to researches[1]. These results show that our recommendation system is working properly.

5.2.6 Comparison of Gradient Descent and Stochastic Gradient Descent

The above shared test results were based on the use of stochastic gradient descent throughout the project. At this point, making a comparison will be helpful in explaining why stochastic gradient descent is used. The word stochastic means probabilistic. In stochastic gradient descent, a few random samples are chosen for each iteration instead of the whole data set. In gradient descent, a whole data set is handled instead of a few random samples. For this reason, reaching the minimum in gradient descent is less noisy and less random manner[16]. Although it sounds good, the cost of typical gradient descent is quite high, and it has more successful results but lose its value considering this cost. Let's say you have a data set of 1 million rows, each iteration until the minimum is reached that you will need to use all of one million samples to complete one iteration while performing the Gradient Descent. Therefore, it becomes very expensive to perform computationally. This problem is solved by stochastic gradient descent[16].

In SGD, it only uses a few random samples to perform each iteration. The sample is randomly mixed and selected to perform the iteration. So, in SGD, instead of the sum of the gradient of the cost function of all instances, we find the gradient of the cost function of a single instance at each iteration. At the same time, while stochastic gradient descent reaches its minimum due to this randomness, it requires more iteration as it reaches in a noisy and random way. Although this is thought to be more cost at first, the training process is actually completed in a much shorter time. Larger deviations and in some cases the high number of errors are ignored because they are realized at much lower cost and stochastic gradient descent is used in large data sets.

The data set also trained using gradient descent. The graph below shows the RMSE error rate of GD. It is trained with the Amazon’s data set as in the stochastic gradient descent.

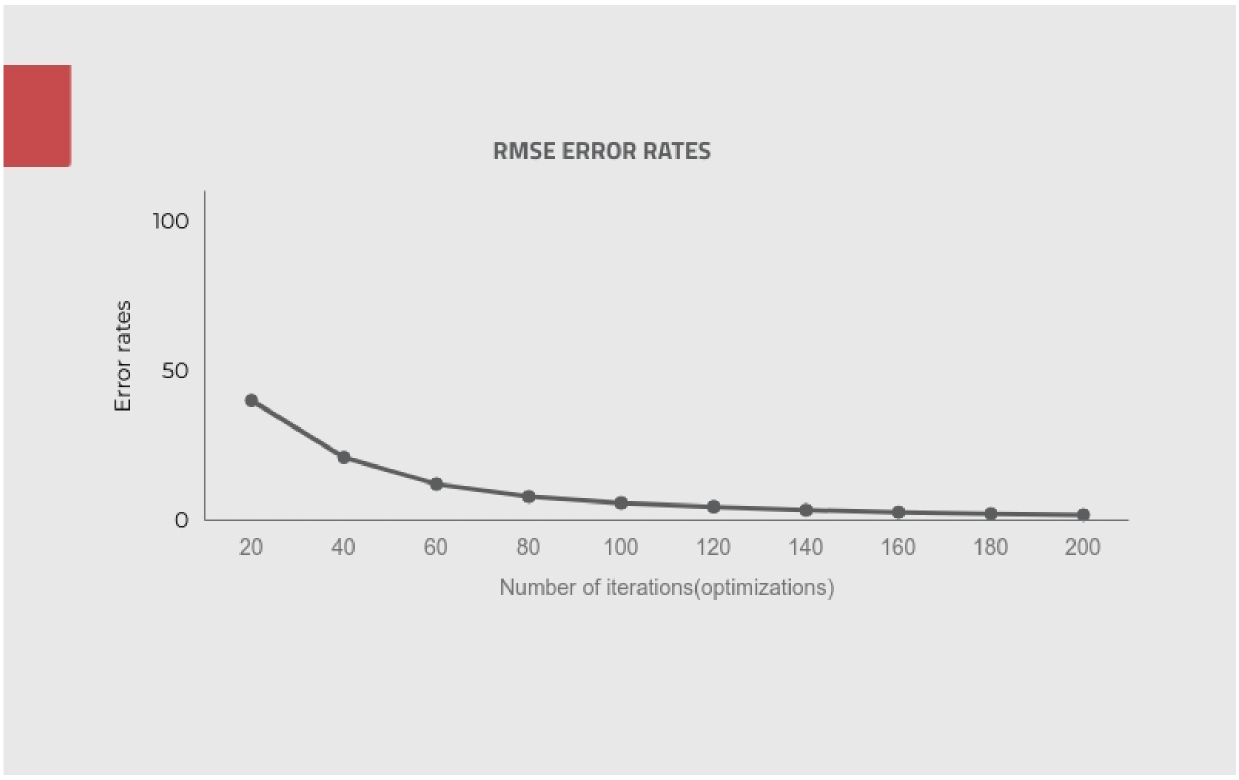


Figure 5.8 RMSE error rates for every 20th iteration in GD

As you can see in the graph above, the error rate of gradient descent in the 20th iteration is much lower than the error rate in Figure 5.7. This is because the whole data set is included in the training process instead of choosing a few examples at the beginning. In the 0th iteration, both start with almost the same error rate because of the randomly assigned hidden property values. Later, we need to pay attention here, when the last iteration is completed, it will be seen that the error rate is less than the stochastic gradient descent. This may at first suggest that gradient descent is more successful. In fact, this idea is quite correct, but the cost of gradient descent is so much higher that it loses its success. Within the scope of this project, it took about 30 minutes to complete 200 iterations with stochastic gradient descent, but this situation takes more than 2 hours for gradient descent. Therefore, although stochastic gradient descent is more prone to error, it is preferred by large companies due to its low cost.

In fact, there are many more recommendation system evaluation metrics [10]. Within the scope of this project, the matrix factorization technique, which won the Netflix competition award in 2006, and the RMSE evaluation metric were used. According to the researches, it is one of the ideal metrics used when evaluating RMSE or MAE recommendation systems. The test part was the last topic of the implementation section, and now we will switch to the conclusion and future work section.

6. Conclusion & Future Work

If we briefly summarize the work done within the scope of the project, first of all, after a great research and knowledge acquisition on recommendation systems, the step of finding a data set was started. In this step, the publicly available data set Amazon obtained from users between 1996 and 2014 was used. In the next step, with the arrangement of this data set in accordance with the recommendation system, the filtering and training part was made. In this section, the mathematics underlying the algorithms used within the scope of the project is explained and how the training process progresses with examples. In the last part, in the evaluation process of the algorithms used within the scope of the project, the very popular matrix factorization method[1] and the RMSE error finding formula were used. As a result, the expectations mentioned in motivation and aims have been achieved.

In the future work, the development of this project by combining it with other filtering techniques in addition to the collaborative filtering technique and creating a hybrid recommendation system can yield more successful results. This is because a hybrid model will consist of more than one technique, so creating a recommendation system by using the strengths of each model and getting rid of its weaknesses can yield good results. There are also many studies and researches on this subject[10].

In addition, the recommendation process and error rates can be more successful by using and training a more comprehensive and regular data set. Of course, this could not be achieved within the scope of this project as much more powerful machines were needed for this, but aiming to advance this process and reinforce my knowledge in the future.

Bibliography

[1] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems."*Computer* 42.8 (2009): 30-37.

[2] Rashid, Al Mamunur, et al. "Getting to know you: learning new user preferences in recommender systems." *Proceedings of the 7th international conference on Intelligent user interfaces*. 2002.

[3] Schafer, J. Ben, Joseph Konstan, and John Riedl. "Recommender systems in e-commerce." *Proceedings of the 1st ACM conference on Electronic commerce*. 1999.

[4] Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." *IEEE transactions on knowledge and data engineering* 17.6 (2005): 734-749.

[5] Ziegler, Cai-Nicolas, et al. "Improving recommendation lists through topic diversification." *Proceedings of the 14th international conference on World Wide Web*. 2005.

[6] Schafer, J. Ben, et al. "Collaborative filtering recommender systems." *The adaptive web*. Springer, Berlin, Heidelberg, 2007. 291-324.

[7] Gábor Takács et al (2008). Matrix factorization and neighbor based algorithms for the Netflix prize problem. In: Proceedings of the 2008 ACM Conference on Recommender Systems, Lausanne, Switzerland, October 23 - 25, 267-274.

[8] Daniel D. Lee and H. Sebastian Seung (2001). Algorithms for Non-negative Matrix Factorization. Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference. MIT Press. pp. 556–562.

[9] Bennett, James, and Stan Lanning. "The netflix prize." *Proceedings of KDD cup and workshop*. Vol. 2007. 2007.

[10] Shani, Guy, and Asela Gunawardana. "Evaluating recommendation systems." *Recommender systems handbook*. Springer, Boston, MA, 2011. 257-297.

[11] Bell, Robert M., and Yehuda Koren. "Lessons from the Netflix prize challenge." *Acm Sigkdd Explorations Newsletter* 9.2 (2007): 75-79.

[12] Levy, Kenneth L., and Neil E. Lofgren. "Recommendation Systems." U.S. Patent Application No. 12/764,091.

[13] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *the Journal of machine Learning research* 12 (2011): 2825-2830.

[14] Pazzani, Michael J., and Daniel Billsus. "Content-based recommendation systems."*The adaptive web*. Springer, Berlin, Heidelberg, 2007. 325-341.

[15] Michie, Donald, David J. Spiegelhalter, and C. C. Taylor. "Machine learning."*Neural and Statistical Classification* 13.1994 (1994): 1-298.

[16] Bottou, Léon. "Stochastic gradient descent tricks."*Neural networks: Tricks of the trade*. Springer, Berlin, Heidelberg, 2012. 421-436.

[17] Fuentes-Fernández, Lidia, and Antonio Vallecillo-Moreno. "An introduction to UML profiles."*UML and Model Engineering* 2.6-13 (2004): 72.